Unit – 1

Key concepts of Object-Oriented Programming –Advantages – Object Oriented Languages – I/O in C++ - C++ Declarations. Control Structures: - Decision Making and Statements: If.. Else, jump, go to, break, continue, Switch case statements-Loops in C++: for, while, do- functions in C++- In line functions–Function Overloading..

Object-Oriented Programming (OOP) is a programming paradigm that uses objects – instances of classes – to organize and structure code. Here are some key concepts of OOP, advantages, and information about Object-Oriented Languages, with a focus on C++:

I) Key Concepts of Object-Oriented Programming:

1. Class:

- A blueprint or template for creating objects.

- Defines attributes (data members) and behaviors (member functions/methods) of objects.

2. Object:

- An instance of a class.

- Represents a real-world entity and encapsulates data and behavior.

3. Encapsulation:

- The bundling of data (attributes) and methods (functions) that operate on the data into a single unit (class).

- Provides data hiding and protection.

4. Inheritance:

- A mechanism that allows a class to inherit properties and behaviors from another class.

- Supports code reusability and the creation of a hierarchy of classes.

5. Polymorphism:

- The ability of a function or method to behave in different ways based on the context.
- Achieved through function overloading and overriding.

6. Abstraction:

- The process of simplifying complex systems by modeling classes based on the essential properties and behaviors they possess.
- Hides the implementation details and focuses on the essential features.

II) Advantages of Object-Oriented Programming:

1. Modularity:

- Code is organized into modular units (classes), making it easier to understand, maintain, and debug.

2. Reusability:

- Classes and objects can be reused in different parts of the program or in other programs, promoting code reuse.

3. Scalability:

- Easily scalable as new classes and objects can be added without affecting existing code.

4. Flexibility:

- Supports changes and updates without affecting the entire system, promoting a flexible and adaptable codebase.

5. Maintenance:

- Easier to maintain and update due to the modular and organized structure.

6. Security:

- Encapsulation helps in data hiding, which enhances security by preventing unauthorized access to data.

Object-Oriented Languages in C++:

C++ is a versatile programming language that supports both procedural and object-oriented programming. Some key features of C++ related to OOP include:

1. Classes and Objects:

- C++ allows the definition of classes and the creation of objects based on those classes.

2. Inheritance:

- C++ supports both single and multiple inheritance, allowing a class to inherit properties from one or more classes.

3. Polymorphism:

- Achieved through function overloading and virtual functions.

4. Encapsulation:

- C++ supports encapsulation by allowing the specification of access modifiers (public, private, protected) for class members.

5. Abstraction:

- Abstraction is achieved through the creation of abstract classes and pure virtual functions.

C++'s support for these OOP concepts makes it a powerful and widely used language for building complex software systems. The combination of procedural and object-oriented features gives developers flexibility in choosing the right paradigm for a particular task.

III) FUNCTIONS OVERLOADING:

Function overloading in C++ allows you to define multiple functions with the same name but with different parameter lists. The compiler determines which function to call based on the number or types of arguments passed to the function. Function overloading provides a way to use the same function name for different functionalities, making the code more readable and expressive.

Ex:

```
#include <iostream>

using namespace std;


// Function with two integer parameters

int add(int a, int b) {

   return a + b;

}


// Function with three double parameters

double add(double a, double b, double c) {

   return a + b + c;

}


// Function with a char and a string parameter

string add(char a, string b) {
```

```
    return a + b;
}

int main() {
    // Calling the first function
    cout << "Sum of integers: " << add(5, 3) << endl;

    // Calling the second function
    cout << "Sum of doubles: " << add(2.5, 3.7, 1.2) << endl;

    // Calling the third function
    cout << "Concatenation: " << add('A', "BCD") << endl;

    return 0;
}
```

In this example:

- The **add** function is overloaded three times with different parameter lists.
- The first version takes two integers and returns their sum.
- The second version takes three doubles and returns their sum.
- The third version takes a char and a string, concatenates them, and returns the result.

When you call the **add** function in the **main** function, the compiler determines which version of the function to invoke based on the arguments you provide.

Function overloading is not limited to the number of parameters; it can also consider the types of parameters. The key is to have different parameter lists for the overloaded functions.